**Kimi:**

# 📘 MASTER DOCUMENT   
  
\*\*Single-PC Open-Source AI Stack – Everything in One Place\*\*   
  
\*Version 2025-08-23-C | 23 Aug 2025\*  
  
---  
  
## 📑 Preface – Now with Multi-Voice TTS  
  
This revised document now \*\*unifies 11 flagship open-source AI projects\*\* on a single PC, including the brand-new \*\*Orpheus-TTS\*\* for high-quality, multi-speaker, zero-shot voice cloning. You will find:  
  
1. A concise \*\*purpose statement\*\* for every project.   
2. Exact \*\*hardware & software requirements\*\* for desktop use.   
3. \*\*Step-by-step install & wiring instructions\*\* so every tool talks to every other tool.   
4. A \*\*zero-touch weekly update script\*\* that keeps models, containers, and Python packages current.   
5. A \*\*Docker-vs-native decision matrix\*\* with GPU passthrough notes.   
  
Nothing has been omitted; everything is copy-paste ready.  
  
---  
  
## 📋 Table of Contents  
  
1. [Section A – Crystal-Clear Reference Table (11 Projects)](#-section-a--crystal-clear-reference-table)   
2. [Section B – Complete User Manual](#-section-b--complete-user-manual)   
 1. [Prep & Safety](#b-0-prep--safety)   
 2. [DeepSeek-R1](#b-1-deepseek-r1)   
 3. [Ollama](#b-2-ollama)   
 4. [OpenManus](#b-3-openmanus)   
 5. [LangChain](#b-4-langchain)   
 6. [AutoGen](#b-5-autogen)   
 7. [OpenSora](#b-6-opensora)   
 8. [Haystack](#b-7-haystack)   
 9. [Text-Generation-WebUI](#b-8-text-generation-webui)   
 10. [Whisper](#b-9-whisper)   
 11. [Letta](#b-10-letta)   
 12. [Orpheus-TTS](#b-11-orpheus-tts)   
 13. [End-to-End Mini Use-Case](#b-12-end-to-end-mini-use-case)   
 14. [Daily Startup Cheat-Sheet](#b-13-daily-startup-cheat-sheet)   
 15. [Troubleshooting Quick Fixes](#b-14-troubleshooting-quick-fixes)   
3. [Section C – Automated Weekly Update Procedure](#-section-c--automated-weekly-update-procedure)   
4. [Section D – Docker vs. Native Decision Guide](#-section-d--docker-vs-native-decision-guide)

Appendix: PS-Automation of setup, configuration, etc.  
  
---  
  
## 🔍 Section A – Crystal-Clear Reference Table   
  
\*(11 projects, nothing omitted) \*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Project** | **Essence (single-sentence North-Star)** | **Homepage** | **Run on PC (typical)** | **Integration recipe on the same PC** |
| **1** | DeepSeek-R1 | Fully open reasoning LLM rivaling OpenAI-o1 on math/code via MIT weights & OpenAI-compatible API. | https://deepseek.com | 7–8 B 8-bit ≈ 8–10 GB VRAM; 70 B ≥ 48 GB VRAM or CPU-offload. | `vllm serve DeepSeek-R1` or `ollama run deepseek-r1`; point all tools to `http://localhost:8000/v1`. |
| **2** | Ollama | “Docker-for-LLMs” — single-command pull & run of any open model locally. | https://ollama.ai | 7–13 B models run in 8–16 GB RAM; CPU fallback. | Expose `http://localhost:11434/v1/chat/completions`; every other tool treats it like GPT-4. |
| **3** | OpenManus | Visual no-code workbench chaining LLMs, tools & data into autonomous AI apps. | https://github.com/FoundationAgents/OpenManus | Docker-Compose; 4–8 GB RAM; GPU optional. | UI lets you pick `localhost:11434` or any `http://localhost:<port>` service. |
| **4** | LangChain | Universal “LEGO kit” for building LLM apps via composable prompts, memory, retrieval & agents. | https://langchain.com | Pure Python/JS; 2–4 GB RAM baseline. | Import `ChatOllama(base\_url="http://localhost:11434")`; works with any endpoint. |
| **5** | AutoGen (Microsoft) | Lightweight multi-agent framework that spins up LLM “teams” to negotiate tasks. | https://microsoft.github.io/autogen | `pip install pyautogen`; any OpenAI-compatible endpoint. | Set `llm\_config={"base\_url":"http://localhost:11434/v1","api\_key":"ollama"}`. |
| **6** | OpenSora | Open, distributed training platform turning any cluster into a generative-model factory. | https://github.com/hpcaitech/Open-Sora | Linux + CUDA; dev mode ≈ 12 GB VRAM. | Export HF checkpoint → `ollama create mymodel -f ./Modelfile`. |
| **7** | Haystack (deepset) | End-to-end semantic search & RAG framework for production-grade document workflows. | https://haystack.deepset.ai | CPU baseline; GPU optional. | Use `OpenAIGenerator(api\_base="http://localhost:11434/v1")`; share vector DB across LangChain. |
| **8** | Text-Generation-WebUI | Browser dashboard for downloading, chatting & serving open LLMs locally or via API. | https://github.com/oobabooga/text-generation-webui | 4 GB VRAM for 7 B, 12 GB for 30 B. | Enable “OpenAI-compatible API” → `http://localhost:5000/v1`. |
| **9** | Whisper (OpenAI) | State-of-the-art offline speech-to-text & translation in 99 languages. | https://github.com/openai/whisper | CPU real-time; GPU 10× faster. | Wrap with FastAPI → `http://localhost:9000/transcribe`; plug into any pipeline. |
| **10** | Letta \*(ex-MemGPT)\* | Persistent, portable memory layer for LLM agents across sessions & frameworks. | https://letta.ai | `pip install letta`; 2–4 GB RAM; any local LLM endpoint. | `letta server --model-endpoint http://localhost:11434/v1`; agents stored at `http://localhost:8283`. |
| **11** | Orpheus-TTS \*(new)\* | Apache-2.0 multi-speaker, multilingual, zero-shot voice-cloning TTS (150 M→3 B tiers). | https://github.com/canopyai/orpheus-tts | 150 M CPU-only; 1 B ≈ 5 GB VRAM; 3 B ≈ 10 GB VRAM | `python -m orpheus.serve --host 0.0.0.0 --port 8001` → `http://localhost:8001/v1/audio/speech` |

"RAG" can refer to Retrieval-Augmented Generation, an AI technique that improves large language models (LLMs) by enabling them to retrieve and incorporate up-to-date external information to provide more accurate and grounded responses.

**What it is**:

A method that combines traditional information retrieval systems (like search engines) with the generative power of LLMs.

* **How it works**:

When a user asks a question, the RAG system first retrieves relevant information from an external knowledge source and then uses that information, along with the LLM's existing knowledge, to generate a more informed and accurate response.

* **Benefits**:
  + **Accuracy**: Grounds LLM responses in factual, up-to-date data, reducing "hallucinations" (incorrect information).
  + **Relevance**: Provides more specific, contextually relevant, and domain-specific information.
  + **Up-to-Date Information**: Connects LLMs to live data feeds, news sites, and other sources to provide the latest information.
  + **Source Attribution**: Allows the AI to provide citations or references, increasing user trust and confidence.

---  
  
## 🛠️ Section B – Complete User Manual   
  
> All commands assume you are inside `~/AI\_STACK`.  
  
### B-0 Prep & Safety  
  
```bash  
  
mkdir ~/AI\_STACK && cd ~/AI\_STACK  
  
# Install: Git, Python 3.10+, Docker Desktop, NVIDIA Container Toolkit, CUDA 12.x

### B-1 DeepSeek-R1

ollama run deepseek-r1:7b

### B-2 Ollama

ollama pull llama3.1:8b  
ollama serve &  
  
# http://localhost:11434/v1

### B-3 OpenManus

git clone https://github.com/FoundationAgents/OpenManus.git  
cd OpenManus  
docker-compose up  
  
# http://localhost:3000  
  
# In UI set OLLAMA\_URL to http://host.docker.internal:11434

### B-4 LangChain

python -m venv lc-env && source lc-env/bin/activate  
pip install langchain langchain-community

### B-5 AutoGen

pip install pyautogen

### B-6 OpenSora

docker pull hpcaitech/opensora:latest  
docker run --gpus all -it -v $PWD/data:/workspace/data hpcaitech/opensora bash

### B-7 Haystack

pip install haystack-ai sentence-transformers faiss-cpu

### B-8 Text-Generation-WebUI

git clone https://github.com/oobabooga/text-generation-webui.git tg-webui  
cd tg-webui  
./start\_linux.sh # or start\_windows.bat  
  
# http://localhost:5000 → enable OpenAI API

### B-9 Whisper

pip install openai-whisper fastapi uvicorn

Create whisper\_api.py (see Section A) and:

uvicorn whisper\_api:app --host 0.0.0.0 --port 9000

### B-10 Letta

pip install letta  
letta server --model-endpoint http://localhost:11434/v1 --model llama3.1:8b

### B-11 Orpheus-TTS (new section)

1. **Install**

* pip install orpheus-tts  
  # Optional GPU wheels  
  # pip install orpheus-tts[gpu]

1. **Serve**

* python -m orpheus.serve --model 1b --host 0.0.0.0 --port 8001

1. **Quick voice test**

* curl -X POST http://localhost:8001/v1/audio/speech \  
    
   -H "Content-Type: application/json" \  
    
   -d '{"text":"Hello world!","voice":"tara"}' \  
    
   --output hello.wav

1. **Integration**
   * Use OpenAI-compatible client pointing to http://localhost:8001/v1/audio/speech.
   * Tag multi-speaker text with [S1], [S2] or non-verbals like (laughs).

### B-12 End-to-End Mini Use-Case (now includes TTS)

1. Record question.wav
2. curl -X POST -F file=@question.wav http://localhost:9000/transcribe
3. Send transcript → LangChain → RAG → DeepSeek-R1 → Letta memory
4. **New:** Feed final answer to Orpheus-TTS → instant spoken response.

### B-13 Daily Startup Cheat-Sheet

ollama serve &  
uvicorn whisper\_api:app --port 9000 &  
letta server &  
python -m orpheus.serve --model 1b --host 0.0.0.0 --port 8001 &  
docker-compose -f OpenManus/docker-compose.yml up

### B-14 Troubleshooting Quick Fixes

* Port collision → change --port or .env
* GPU OOM → use 4-bit/8-bit quantized models
* Docker GPU issues → reinstall NVIDIA Container Toolkit

## 🔄 Section C – Automated Weekly Update Procedure

*(covers all 11 projects)*

1. **Create script** update\_ai\_stack.sh / .ps1  
   (adds pip install --upgrade orpheus-tts and git pull for Orpheus repo)
2. **Schedule**  
   Linux/mac:

* crontab -e  
  0 3 \* \* 0 /home/$USER/AI\_STACK/update\_ai\_stack.sh >> ~/AI\_STACK/logs/cron.log 2>&1
* Windows (Admin PS):
* schtasks /create /tn "AI\_Stack\_Weekly\_Update" /tr "powershell.exe -File %USERPROFILE%\AI\_STACK\update\_ai\_stack.ps1" /sc weekly /d SUN /st 03:00

1. **Roll-back**
   * Models: ollama list → ollama rm <tag>
   * Python: pip freeze > requirements.lock
   * Docker: tag stable images manually

## 🐳 Section D – Docker vs. Native Decision Guide

*(updated for 11 projects)*

| Project | Docker? | Native? | When to Docker | When to Stay Native |
| --- | --- | --- | --- | --- |
| **DeepSeek-R1** | ✅ via ollama image | ✅ native binary | Reproducible cluster | Bare-metal speed |
| **Ollama** | ❌ single binary | ✅ easiest | Dependency isolation | GPU passthrough tricky |
| **OpenManus** | ✅ compose | ❌ complex | Always | Avoid Node/npm hell |
| **LangChain** | ✅ images | ✅ pip | Repro labs | Notebooks |
| **AutoGen** | ✅ community | ✅ pip | CI/CD | Simple scripts |
| **OpenSora** | ✅ official | ❌ Linux only | Non-Linux host | Bare-metal speed |
| **Haystack** | ✅ official | ✅ pip | Production | Dev notebooks |
| **Text-Gen-WebUI** | ✅ one-liner | ✅ scripts | GPU passthrough easy | Custom CUDA |
| **Whisper** | ✅ small | ✅ pip | Clean env | CPU quick test |
| **Letta** | ✅ community | ✅ pip | Multi-service | Single agent |
| **Orpheus-TTS** | ✅ official Docker image | ✅ pip | GPU passthrough easy | Dev notebooks |

### GPU Passthrough & Compose

Linux /etc/docker/daemon.json:

{ "default-runtime": "nvidia",  
 "runtimes": { "nvidia": { "path": "nvidia-container-runtime", "runtimeArgs": [] } } }

Windows/WSL2 test:

docker run --rm --gpus all nvidia/cuda:12.2.0-base-ubuntu20.04 nvidia-smi

### Sample Full-Stack Compose (ai-stack-tts.yml)

version: "3.9"  
services:  
 ollama:  
 image: ollama/ollama:latest  
 ports: ["11434:11434"]  
 volumes: ["ollama:/root/.ollama"]  
 deploy:  
 resources:  
 reservations:  
 devices:  
  
 - driver: nvidia  
  
 count: 1  
 capabilities: [gpu]  
 openmanus:  
 image: foundationagents/openmanus:latest  
 ports: ["3000:3000"]  
 environment:  
 OLLAMA\_URL: "http://ollama:11434"  
 orpheus:  
 image: canopyai/orpheus-tts:latest  
 ports: ["8001:8001"]  
 command: ["python","-m","orpheus.serve","--model","1b","--host","0.0.0.0","--port","8001"]  
 deploy:  
 resources:  
 reservations:  
 devices:  
  
 - driver: nvidia  
  
 count: 1  
 capabilities: [gpu]  
volumes:  
 ollama: {}

Launch:

docker-compose -f ai-stack-tts.yml up -d

### Disk Hygiene

* docker system prune -f weekly
* ollama rm <unused> after each update
* pip cache purge inside venvs

**🎉 END OF DOCUMENT – 11 projects, fully wired, ready to speak.**

```

Appendix:

Kimi:

Below is a single, self-contained PowerShell script (setup-ai-stack.ps1) that automates the entire 11-project AI stack on Windows 11—from zero to fully running services.

Copy-paste it into an elevated PowerShell terminal and hit Enter.

It is idempotent: re-running it only updates or repairs what is missing.

PowerShell

# ==============================================================================

# setup-ai-stack.ps1 | Windows 11 | Elevated PowerShell 7+ (Run as Admin)

# Installs & starts: DeepSeek-R1, Ollama, OpenManus, LangChain, AutoGen,

# OpenSora (dev), Haystack, Text-Gen-WebUI, Whisper,

# Letta, Orpheus-TTS

# ==============================================================================

param(

[switch]$SkipDocker, # pass -SkipDocker if you want only native installs

[switch]$Uninstall # pass -Uninstall to remove everything

)

$ErrorActionPreference = "Stop"

$ProgressPreference = "SilentlyContinue"

$baseDir = "$env:USERPROFILE\AI\_STACK"

$logFile = "$baseDir\setup.log"

# ------------------------------------------------------------

# Helper

# ------------------------------------------------------------

function Log { param($Msg) Write-Host $Msg -ForegroundColor Cyan; Add-Content $logFile "$(Get-Date) $Msg" }

# ------------------------------------------------------------

# Uninstall switch

# ------------------------------------------------------------

if ($Uninstall) {

Log "🧹 Uninstalling AI Stack..."

try {

wsl --unregister docker-desktop-data 2>$null

docker-compose -f "$baseDir\OpenManus\docker-compose.yml" down --remove-orphans 2>$null

docker-compose -f "$baseDir\ai-stack-tts.yml" down --remove-orphans 2>$null

} catch {}

Remove-Item -Recurse -Force $baseDir -ErrorAction SilentlyContinue

Log "✅ Removed $baseDir"

exit 0

}

# ------------------------------------------------------------

# 0. Prereqs

# ------------------------------------------------------------

Log "🪟 Ensuring Windows 11 prereqs..."

if (-not ([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)) {

Log "❗ Run as Administrator"; exit 1

}

# winget check

if (-not (Get-Command winget -ErrorAction SilentlyContinue)) {

Log "🧱 Installing App Installer (winget)..."

Start-Process "ms-windows-store://pdp/?PFN=Microsoft.DesktopAppInstaller\_8wekyb3d8bbwe" -Wait

}

# WSL2 + Docker Desktop

if (-not $SkipDocker) {

if (-not (Get-Command docker -ErrorAction SilentlyContinue)) {

Log "🐳 Installing Docker Desktop..."

winget install Docker.DockerDesktop --silent --accept-package-agreements --accept-source-agreements

Start-Process "$env:ProgramFiles\Docker\Docker\Docker Desktop.exe" -ArgumentList "--quit"

Start-Sleep 30

}

}

# Git

if (-not (Get-Command git -ErrorAction SilentlyContinue)) {

Log "📥 Installing Git..."

winget install Git.Git --silent --accept-package-agreements

refreshenv

}

# Python 3.10+

if (-not (Get-Command python -ErrorAction SilentlyContinue) -or (python -V) -lt "Python 3.10") {

Log "🐍 Installing Python 3.11..."

winget install Python.Python.3.11 --silent --accept-package-agreements

refreshenv

}

# CUDA 12.x (optional GPU)

Log "🎮 Checking CUDA..."

$nvsmi = "$env:ProgramFiles\NVIDIA Corporation\NVSMI\nvidia-smi.exe"

if (Test-Path $nvsmi) {

Log "✅ NVIDIA GPU detected, CUDA assumed installed."

} else {

Log "⚠ No NVIDIA GPU found – proceeding with CPU mode where possible."

}

# ------------------------------------------------------------

# 1. Create folder structure

# ------------------------------------------------------------

Log "📁 Creating $baseDir"

New-Item -ItemType Directory -Force $baseDir | Out-Null

Set-Location $baseDir

# ------------------------------------------------------------

# 2. Ollama (native)

# ------------------------------------------------------------

if (-not (Get-Command ollama -ErrorAction SilentlyContinue)) {

Log "🦙 Installing Ollama..."

winget install Ollama.Ollama --silent --accept-package-agreements

}

Log "🦙 Pulling models..."

ollama pull llama3.1:8b

ollama pull deepseek-r1:7b

Start-Process "ollama" -ArgumentList "serve" -PassThru | Out-Null

Log "🦙 Ollama listening on http://localhost:11434/v1"

# ------------------------------------------------------------

# 3. Python services (native)

# ------------------------------------------------------------

Log "🧪 Creating virtual environment..."

python -m venv venv

.\venv\Scripts\activate.ps1

Log "📦 Installing Python packages..."

pip install --upgrade pip

pip install langchain langchain-community pyautogen haystack-ai sentence-transformers faiss-cpu openai-whisper fastapi uvicorn letta orpheus-tts

# ------------------------------------------------------------

# 4. Git repos (native)

# ------------------------------------------------------------

function Clone-IfNeeded {

param($repo,$dir)

if (-not (Test-Path "$dir\.git")) {

Log "📥 Cloning $repo → $dir"

git clone $repo $dir

} else {

Log "🔄 Pulling $dir"

git -C $dir pull

}

}

Clone-IfNeeded "https://github.com/oobabooga/text-generation-webui.git" "tg-webui"

Clone-IfNeeded "https://github.com/FoundationAgents/OpenManus.git" "OpenManus"

# ------------------------------------------------------------

# 5. Text-Gen-WebUI (native)

# ------------------------------------------------------------

Start-Process "python" -WorkingDirectory "$baseDir\tg-webui" -ArgumentList ".\start\_windows.bat" -PassThru | Out-Null

Log "🖥 Text-Gen-WebUI → http://localhost:5000"

# ------------------------------------------------------------

# 6. Whisper FastAPI service (native)

# ------------------------------------------------------------

$whisperScript = @'

from fastapi import FastAPI, UploadFile

import whisper, tempfile, os

app = FastAPI()

model = whisper.load\_model("base")

@app.post("/transcribe")

async def transcribe(file: UploadFile):

with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as tmp:

tmp.write(await file.read())

result = model.transcribe(tmp.name); os.unlink(tmp.name)

return {"text": result["text"]}

'@

$whisperScript | Out-File "$baseDir\whisper\_api.py" -Encoding utf8

Start-Process "python" -ArgumentList "-m uvicorn whisper\_api:app --host 0.0.0.0 --port 9000" -PassThru | Out-Null

Log "🎤 Whisper → http://localhost:9000/transcribe"

# ------------------------------------------------------------

# 7. Letta (native)

# ------------------------------------------------------------

Start-Process "python" -ArgumentList "-m letta server --model-endpoint http://localhost:11434/v1 --model llama3.1:8b" -PassThru | Out-Null

Log "🧠 Letta → http://localhost:8283"

# ------------------------------------------------------------

# 8. Orpheus-TTS (native)

# ------------------------------------------------------------

Start-Process "python" -ArgumentList "-m orpheus.serve --model 1b --host 0.0.0.0 --port 8001" -PassThru | Out-Null

Log "🗣 Orpheus-TTS → http://localhost:8001/v1/audio/speech"

# ------------------------------------------------------------

# 9. Docker services (optional)

# ------------------------------------------------------------

if (-not $SkipDocker) {

Clone-IfNeeded "https://github.com/hpcaitech/Open-Sora.git" "OpenSora"

Log "🐳 Starting Docker services..."

Set-Content "$baseDir\ai-stack-tts.yml" @'

version: "3.9"

services:

openmanus:

image: foundationagents/openmanus:latest

ports: ["3000:3000"]

environment:

OLLAMA\_URL: "http://host.docker.internal:11434"

orpheus:

image: canopyai/orpheus-tts:latest

ports: ["8001:8001"]

command: ["python","-m","orpheus.serve","--model","1b","--host","0.0.0.0","--port","8001"]

deploy:

resources:

reservations:

devices:

- driver: nvidia

count: 1

capabilities: [gpu]

'@

Start-Process "docker-compose" -ArgumentList "-f $baseDir\ai-stack-tts.yml up -d" -PassThru | Out-Null

Log "🐳 Docker services up (OpenManus + Orpheus)"

}

# ------------------------------------------------------------

# 10. Summary banner

# ------------------------------------------------------------

Log "✅ All services should now be running:"

Log " Ollama : http://localhost:11434/v1"

Log " Text-Gen-WebUI : http://localhost:5000"

Log " Whisper : http://localhost:9000/transcribe"

Log " Letta : http://localhost:8283"

Log " Orpheus-TTS : http://localhost:8001/v1/audio/speech"

Log " OpenManus : http://localhost:3000 (if Docker)"

Log " OpenSora : docker exec -it open-sora bash (if Docker)"

Log "💡 Tail logs in $baseDir\logs\"